Label Optimal Regret Bounds for Online Local Learning

Kevin A. Lai

A Thesis
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance
by the Department of
Computer Science

Adviser: Moses Charikar

June 2015

Abstract

In many settings, we would like to make predictions about the local structure of a problem rather than explicitly trying to learn some global structure. For instance, given two teams from some set of teams, we may want to know if one team will beat the other, while we may not be explicitly interested in a ranking of all of the teams. A recent framework proposed by Christiano (2014a) called "online local learning" captures this broad class of problems, which includes online max cut and online gambling.

In the online local learning setting, we receive pairs of items from some set of items of size n, and we must label them from some set of labels of size l. Christiano (2014a) provides a Follow-the-Regularized-Leader algorithm using a log determinant regularizer that obtains regret $O(\sqrt{nl^3T})$. In this thesis, we improve the analysis of Christiano's algorithm to obtain regret $O(\sqrt{nlT})$. We also present a matching lower bound based on a reduction from the planted clique problem, which is believed to have no polynomial-time algorithm for planted cliques of size $k = o(n^{1/2})$. We then show a similar reduction from the planted dense subgraph problem, which is believed to be even harder than the planted clique problem. This thesis is a more detailed treatment of the material from Awasthi et al. (2015), which first showed these results.

Acknowledgements

This thesis is based on joint work with Andrej Risteski, Pranjal Awasthi, and Moses Charikar. I'd like to thank my collaborators for making this work possible. I would like to thank Moses in particular for advising me throughout my time at Princeton. I am also very grateful to my parents for always supporting me in my studies. Finally, I would like to thank my friends at Princeton for many illuminating discussions and happy memories.

Contents

Chapter 1

Introduction

In many real-world problems, decisions need to be made on new data or inputs as they arrive. For instance, a router directing traffic on a network may need to route each request through the network as it arrives, while still attempting to keep overall network congestion low. This is an example of an *online* problem. The router's task would be simpler in the *offline* case, in which all data is received in advance of making any decisions. In the offline version of the router problem, the router would see all future requests before deciding how to route each one, whereas in the online version, the router must make a decision immediately after receiving each request.

In *online learning*, a learner plays a series of rounds in which data points are provided one at a time. In each round, the learner must make some prediction about the point she receives, after which she receives some bounded loss function that determines how good her prediction was. In general, the data points and loss functions are provided adversarially, meaning it would be impossible for the learner to obtain any meaningful absolute performance guarantees in the worst case. Instead, we consider some comparison class of hypotheses such that for any given instance of the online learning problem, at least one of the hypothesis in the class will do

well. We then try to bound the worst case *regret*, which is the difference between the learner's performance and the performance of the best hypothesis for a given instance of a problem. Since the data points and loss functions are chosen adversarially in the online setting, online algorithms have robust bounds on performance, which enables them to be used as black boxes in other algorithms. Additionally, many classical algorithms problems have interesting online counterparts.

Much work has been done on online learning problems in the past several decades, and some surprisingly powerful results have been shown. For instance, when the size of the comparison class is small, the *multiplicative weights* algorithm, also called the *experts* algorithm, can give optimal regret in polynomial time (Littlestone and Warmuth, 1994). This algorithm comes from the problem of learning with expert advice, in which a learner must predict each day whether a stock will go up or down. The learner has access to a panel of experts, who each make their own predictions every day. The learner uses this information to make her own prediction, after which she receives a loss function that reveals the "correct" answer (i.e. whether the stock went up or down). The game proceeds for T rounds, and the learner's goal is to do nearly as well as the best expert over all T rounds, even though the learner does not know which expert is best until after all of the rounds have finished. If the number of experts is N, the weighted majority algorithm achieves regret $O(\sqrt{T \log N})$. Moreover, this matches information theoretic lower bounds.

Even though the experts algorithm gives optimal regret, in many cases it is impossible to apply it efficiently because the number of experts would be too large or infinite. One such general setting is the online convex optimization setting. In this setting, in each round the learner must predict some point in a convex compact body $\mathcal{K}$, after which she receives a bounded loss function. The learner competes with the best point over a series of T rounds. For such problems, a powerful efficient meta-

algorithm exists, called Follow-the-Regularized-Leader (Kalai and Vempala, 2005). Follow-the-Regularized-Leader encompasses many algorithms, such as some cases of the experts algorithm and gradient descent, and it provides good regret bounds when applied with a properly chosen regularizer function. We discuss online convex optimization and the Follow-the-Regularized-Leader algorithm in Section 2.1.

A related setting is that of online local learning, in which the learner needs to make predictions about local properties of some set of items. Examples of such problems include online max cut and online gambling. More specifically, the learner receives pairs of items from some set of items of size N and must label them from some set of labels of size l over a series of rounds. We compare the learner's performance to the performance of the best fixed labeling of the items. The intuition for this setting is that since in many cases we only need to give answers about pairs of items, we may be able to find low regret algorithms that do not explicitly compute some consistent labeling for all of the items, which could be computationally infeasible. Note that the experts algorithm could give regret $O(\sqrt{N \log(l) T})$ by simply creating l^N experts: one for each possible labeling of the n items. However, such an algorithm would run in exponential time. The online local learning setting thus tries to capture another scenario in which the experts algorithm fails.

Christiano (2014a) proposes the online local learning setting and gives an algorithm that obtains regret $O(\sqrt{Nl^3T})$. He uses a Follow-the-Regularized-Leader approach with a log determinant regularizer. In this thesis, we review the Follow-the-Regularized-Leader algorithm and the online local learning setting. We then prove the following regret bound for Christiano's algorithm:

Theorem 2 *(Informal).* *For online local learning problems with N items and l labels, there exists an algorithm that achieves $O(\sqrt{NlT})$ regret.*

We will also prove two lower bounds for online local learning algorithms. We base them on the hardness of the planted clique and planted dense subgraph problems for certain parameter regimes. Both of these problems involve distinguishing between an Erdős-Renyi random graph on n nodes and an Erdős-Renyi random graph on n nodes that has a particular subgraph randomly planted in it. We then prove the following theorems:

Theorem 6 *(Informal).* *If there exists a polynomial time online local learning algorithm that achieves $O(\sqrt{Nl^{1-\alpha}T})$ regret for some $0 < \alpha < 1$, then there exists a polynomial time algorithm to solve the planted clique problem for cliques of size $k = o(\sqrt{n})$ with probability $\frac{4}{5}$.*

Theorem 7 *(Informal).* *If there exists a polynomial time algorithm that obtains regret $O(\sqrt{Nl^{\beta}T})$, where $\beta < 1$ is some specific function of the parameters of the problem, then one can solve the planted dense subgraph problem with probability $\frac{4}{5}$ in polynomial time for parameter regimes conjectured to have no polynomial time algorithms.*

Finally, we will compare the online local learning setting to a related setting proposed by Hazan et al. (2012) that achieves better regret bounds for certain problems. We also discuss open problems related to the online local learning setting.

Chapter 2

Upper bound for online local learning

In this section, we show how to model online local learning as an online convex optimization problem. This allows us to use the Follow-the-Regularized-Leader algorithm as a black box, which gives good regret bounds as shown by a theorem from Hazan (2009). We will use a log determinant regularizer for our algorithm, which is the same as was used by Christiano (2014a). However, we improve their analysis of the log determinant regularizer by doing a direct calculation of the Hessian. This leads us to our upper bound on achievable regret for online local learning.

2.1 Online Convex Optimization and Follow-the-Regularized-Leader

Online convex optimization (OCO) covers a broad class of problems in online learning. We follow the OCO framework from Hazan (2009). In that framework, a learner plays a series of T rounds, where in each round t, the learner outputs some x_t from some

convex compact body $\mathcal{K}$. The learner then receives a loss function f_t bounded in [-1,1] and loss $f_t(x_t)$. Since the loss functions are adversarially chosen, the optimum point in each round may shift greatly. As such, we cannot hope to compete against the best possible point for each round. Instead, we compete with the point $x^* \in \mathcal{K}$ that is the best overall in hindsight, that is:

$$x^* = \arg\min_{x \in \mathcal{K}} \sum_{t=1}^{T} f_t(x)$$

The learner seeks to minimize regret, which is the expected difference in loss between the points she plays $\{x_1, ..., x_T\}$ and x^*:

$$\text{Regret} = \mathrm{E}\left[\sum_{t=1}^{T} f_t(x_t)\right] - \sum_{t=1}^{T} f_t(x^*) \tag{2.1.1}$$

where the expectation is over any randomness that the learner uses.

One powerful meta-algorithm for OCO is Follow-the-Regularized-Leader (FTRL), shown in Algorithm 1. In each round t, the FTRL algorithm plays the point that optimizes over the loss functions $f_1, ..., f_{t-1}$ plus an added regularization term $\mathcal{R}(x)$ for some strongly convex function $\mathcal{R}$ called a regularizer. The regret for the FTRL algorithm is given in Theorem 1.

Algorithm 1: Follow-the-Regularized-Leader

Input: $\eta > 0$, convex compact body $\mathcal{K}$, strongly convex regularizer $\mathcal{R}$
$x_1 \leftarrow \arg\min_{x \in \mathcal{K}} \mathcal{R}(x)$
for $t \leftarrow 1$ **to** T **do**
  Play x_t
  Receive loss $f_t(x_t)$
  $x_{t+1} \leftarrow \arg\min_{x \in \mathcal{K}} \sum_{i=1}^{t} f_i(x) + \frac{1}{\eta}\mathcal{R}(x)$
end

Theorem 1. *(Hazan, 2009) Given a convex compact $\mathcal{K}$ and strongly convex regularizer $\mathcal{R}$, define*

$$\lambda = \max_{t,x \in \mathcal{K}} f_t^\intercal [\nabla^2 \mathcal{R}(x)]^{-1} f_t \quad , \quad D = \max_{x_1, x_2 \in \mathcal{K}} |\mathcal{R}(x_1) - \mathcal{R}(x_2)|$$

where $\nabla^2 \mathcal{R}(x)$ is the Hessian of $\mathcal{R}(x)$. Then the FTRL algorithm achieves regret at most $O(\sqrt{\lambda D T})$.

The λ parameter in Theorem 1 corresponds to the strong convexity of the regularizer, measured in the norm of the loss functions. If the regularizer is more strongly convex, λ will be smaller (giving lower regret). The D parameter corresponds to the range of the regularizer over points in $\mathcal{K}$. We refer the reader to Hazan (2009) for a full proof of Theorem 1.

The motivation for FTRL comes from some observed shortcomings of the Follow-The-Leader (FTL) algorithm, which is the same as Algorithm 1, except that we don't use a regularizer–that is, that we set $x_{t+1} = \arg\min_{x \in \mathcal{K}} \sum_{i=1}^{t} f_i(x)$. Essentially, FTL follows the estimated "best" solution for each case, based on the past loss functions. FTL achieves $\Omega(T/2)$ regret in the worst case, which occurs when the "best" solution changes greatly in each round. The regularizer term causes the expression being optimized to be strongly convex, intuitively causing the algorithm to change its guesses more smoothly. Another interpretation for the regularizer is as a "noise" term, which again prevents the algorithm from switching its answer too dramatically.

2.2 Online local learning

In many online learning problems, the learner's performance is compared to some comparison class that represents some global structure in the problem. However, in

each round, the learner may only need to make predictions about local structure in the problem. Thus, while it may be intractable to efficiently reconstruct the global structure, the learner may still achieve low regret by maintaining only local information. This idea is the motivation for the online local learning (OLL) setting of Christiano (2014a).

In the OLL setting, the task is to label pairs of items from some set over a series of rounds. Formally, we have some set of labels $L = \{1, ..., l\}$ and some set of items $S = \{x_1, ..., x_n\}$. In each round, we are presented with a pair of items $(x_i, x_j) \in S \times S$ and are asked to provide some joint distribution p_t over $L \times L$, where t is the current round. We then receive a loss vector $f_t \in [-1, 1]^{l^2}$, and the following loss:

$$\text{Loss}_t(p) = \sum_{a,b \in L} f_t(a, b)p(a, b) \tag{2.2.1}$$

We play this for T rounds. As in the OCO setting, we cannot hope to compete with the best decision in each round. Instead, we will compete with the best fixed labeling in hindsight. A fixed labeling Q corresponds to some mapping $q : S \to L$. When queried with a pair of vertices (x_i, x_j), Q will play a distribution that has probability 1 on the labels $(q(x_i), q(x_j))$ and probability 0 elsewhere. The regret is the difference between our loss and the loss of the best fixed labeling Q, namely:

$$\text{Regret} = \sum_{t=1}^{T} \text{Loss}_t(p_t) - \sum_{t=1}^{T} \text{Loss}_t(Q) \tag{2.2.2}$$

2.3 Label optimal algorithm for online local learning

Christiano (2014a) models OLL as an OCO problem, which he solves with a FTRL algorithm. To do this, he considers the convex set $\mathcal{K}$[1] of $nl \times nl$ positive semidefinite matrices A indexed by object-label pairs such that all entries of A are nonnegative and for all $x_i, x_j \in S$, the following holds:

$$\sum_{a,b \in L} A_{(x_i,a),(x_j,b)} = 1 \tag{2.3.1}$$

That is, for all $x_i, x_j \in S$, the function $p_{x_i,x_j}(a,b) = A_{(x_i,a),(x_j,b)}$ defines a valid probability distribution.

For a given round t, suppose the learner is provided with objects $(x_i, x_j) \in S \times S$ and the learner plays some $A \in \mathcal{K}$. Then the loss vector $f_t \in [-1,1]^{(nl)^2}$ is defined to be zero everywhere except for possibly in the $l \times l$ block corresponding to (x_i, x_j). To compute the loss, we simply treat A as a vector, which gives the following:

$$\text{Loss}_t(A_t) = \sum_{a,b} A_{(x_i,a),(x_j,b)} f_t((x_i,a),(x_j,b)) \tag{2.3.2}$$

We will use $\mathcal{R}(\mathcal{A}) = \log\det(lA + I)$ in this section to denote the log determinant regularizer. This regularizer was chosen because it is the entropy of a Gaussian whose second-order moments are given by A. The Gaussian is the maximum entropy distribution whose second-order moments are given by A, so the log determinant of A gives an upper bound on the entropy of a distribution whose second-order moments

[1]From this point on, $\mathcal{K}$ will refer to the specific body we define here.

are given by A. Such an entropy function will intuitively maximize how smoothly the algorithm changes its guesses over the course of the OLL game.

Christiano (2014a) uses this regularizer to solve OLL using FTRL, as shown in Algorithm 2, achieving a regret bound of $O(\sqrt{nl^3T})$.[2] To show this regret, he gives bounds on the D and λ parameters in Theorem 1. He shows that $D \leq nl$ and $\lambda \leq l^2$. For completeness, we repeat the proof for the bound on D in Appendix A, where we also sketch his bound for λ.

Algorithm 2: OLL algorithm using FTRL

Input: $\eta > 0$, convex compact body $\mathcal{K}$, strongly convex regularizer $\mathcal{R}$
$A^{(1)} \leftarrow \arg\min_{A \in \mathcal{K}} \mathcal{R}(A)$
for $t \leftarrow 1$ **to** T **do**
  Receive (x_i^t, x_j^t)
  Predict (a, b) with probability $A^{(t)}_{(x_i^t, a), (x_j^t, b)}$
  Receive loss function g_t over $L \times L$
  Set $f_t((x_u, c), (x_v, d)) = \begin{cases} g_t(c, d) & \text{if } (x_u, x_v) = (x_i^t, x_j^t) \\ 0 & \text{o.w.} \end{cases}$
  $A^{(t+1)} \leftarrow \arg\min_{A \in \mathcal{K}} \sum_{i=1}^{t} f_i(A) + \frac{1}{\eta}\mathcal{R}(A)$
end

We improve the analysis in Christiano (2014a) to show the following:

Theorem 2. *Follow-the-regularized leader over $\mathcal{K}$ using $\mathcal{R}(\mathcal{A})$ as the regularizer achieves regret $O(\sqrt{nlT})$.*

To prove Theorem 2, we show an improved analysis of the strong convexity parameter of the regularizer. In particular, we show the following:

Lemma 3. *For the log determinant regularizer $\mathcal{R}$ over $\mathcal{K}$, we have $\lambda \leq 4$, where*

$$\lambda = \max_{t, A} f_t^\intercal [\nabla^2 \mathcal{R}(A)]^{-1} f_t$$

[2]For convenience of notation, in Algorithm 2 we let f(A) correspond to the dot product between f and A, where A is treated as an $(nl)^2$ dimensional vector.

as defined in Theorem 1.

We use the following bound from Christiano (2014a) on D:

Lemma 4. *For the log determinant regularizer $\mathcal{R}$ over $\mathcal{K}$, we have $D \leq nl$, where*

$$D = \max_{x_1, x_2 \in \mathcal{K}} |\mathcal{R}(x_1) - \mathcal{R}(x_2)|$$

as defined in Theorem 1.

Proof of Theorem 2. By Lemma 4, the diameter D for $\mathcal{R}$ is at most nl. Then since $\lambda \leq 4$ by Lemma 3, Theorem 1 gives:

$$Regret \leq O(\sqrt{D\lambda T}) = O(\sqrt{nlT}) \tag{2.3.3}$$

□

Before we prove Lemma 3, we need the following lemma, which gives the form of the inverse of the Hessian of $\mathcal{R}(A)$. We prove the lemma in Appendix A.

Lemma 5. *The inverse Hessian H^{-1} of the log determinant regularizer has the following form:*

$$H^{-1}_{(w,x),(y,z)} = -l^{-2}(lA + I)_{y,x}(lA + I)_{w,z}$$

for all $x, w, y, z \in S \times L$.

Proof of Lemma 3. We now prove the bound on $\lambda = \max_{t,A} f_t^\intercal [\nabla^2 \mathcal{R}(A)]^{-1} f_t$. We know that each f_t is nonzero in only l^2 locations, corresponding to the $l \times l$ block of labels for some pair of items $i, j \in S$. Since each entry of f_t is in $[-1, 1]$, it suffices to

bound $\sum_{a,b,c,d\in L} |[\nabla^2\mathcal{R}(A)]^{-1}_{((i,a),(j,b)),((i,c),(j,d))}|$ for all $i, j \in S$. Using Lemma 5, we get:

$$\sum_{a,b,c,d\in L} \left|[\nabla^2\mathcal{R}(A)]^{-1}_{((i,a),(j,b)),((i,c),(j,d))}\right| = \sum_{a,b,c,d\in L} \left|-l^{-2}(lA+I)_{(i,c),(j,b)}(lA+I)_{(i,a),(j,d)}\right| \tag{2.3.4}$$

$$= \frac{1}{l^2}\sum_{b,c\in L}(lA+I)_{(i,c),(j,b)}\sum_{a,d\in L}(lA+I)_{(i,a),(j,d)} \tag{2.3.5}$$

$$= \frac{1}{l^2}\left(\sum_{a,b\in L}(lA+I)_{(i,a),(j,b)}\right)^2 \tag{2.3.6}$$

Then we note the following:

$$\sum_{a,b\in L}(lA+I)_{(i,a),(j,b)} = l\sum_{a,b\in L}A_{(i,a),(j,b)} + \sum_{a,b\in L}I_{(i,a),(j,b)} \tag{2.3.7}$$

$$= l + l = 2l \tag{2.3.8}$$

where we have used (2.3.1) and the definition of the identity. Plugging this into (2.3.6) gives the lemma. □

Chapter 3

Lower bounds for online local learning

We provide computational lower bounds for regret in the OLL setting. In particular, we will show reductions from the planted clique and planted dense subgraph problems to OLL. Both of these problems are conjectured to have no polynomial time algorithms, and we present current conjectures on the hardness of each problem, as well as justifications for the conjectures.

In the OLL setting, one can obtain an information theoretic lower bound by applying the experts algorithm with one expert for each labeling of n items with l labels. This will take exponential time and will achieve regret $O(\sqrt{n \log(l) T})$. We improve this lower bound to show that no polynomial time algorithm can achieve regret $\Omega(\sqrt{n l^{1-\alpha} T})$ for $\alpha > 0$, provided that planted clique or planted dense subgraph have no efficient algorithms for certain parameter regimes.

3.1 Planted clique

In the planted clique problem, we are given some graph G and must distinguish whether it is an Erdős-Renyi random graph $G(n, 1/2)$ or an Erdős-Renyi random graph $G(n, 1/2)$ with a clique of size k planted on a random set of k vertices (i.e. we add all edges between those vertices). We denote the latter graph by $G(n, 1/2, k)$.

The planted clique problem was first proposed by Jerrum (1992) and Kucera (1995). Alon et al. (1988) give a simple spectral algorithm to recover a planted clique for $k = \Omega(n^{1/2})$. Their algorithm considers a set S that consists of the k entries of largest magnitude of the second eigenvector ν_2 of the adjacency matrix. The algorithm finds all vertices with $3k/4$ neighbors in S and outputs those vertices as the planted clique. The algorithm works because with high probability, ν_2 has large magnitude on many vertices of the planted clique. More recently, Dekel et al. (2014) give a linear time algorithm for finding cliques of size $k = \Omega(n^{1/2})$.

For planted cliques of size $k = o(n^{1/2})$, the best known algorithm runs in time $n^{O(\log n)}$, provided that $k \geq 2 \log n$. The algorithm simply iterates over all subsets of vertices of size $2 \log n$. For each subset S that is a clique, the algorithm considers the set of all vertices that are neighbors with every vertex in S. One of these sets of vertices will be the planted clique (Feldman et al., 2013a). Given that this is the state of the art for planted clique algorithms, we have the following conjecture:

Conjecture 1 (Planted clique hardness)**.** *Given a graph G sampled either from $G(n, 1/2)$ or $G(n, 1/2, n^{1/2-\epsilon})$) for any $\epsilon > 0$, there is no polynomial time algorithm that can decide which ensemble G was sampled from with probability $\frac{4}{5}$.*[1]

Feldman et al. (2013a) show that a broad class of algorithms known as statistical algorithms, which include many common algorithms such as PCA and k-means,

[1]The constant is arbitrary and need only be bounded away from $\frac{1}{2}$.

cannot do better than this bound. Meka et al. (2015) show that a constant number of rounds of the Sum-Of-Squares hierarchy cannot find a planted clique of size $k = n^{o(1)}$. These results support the use of planted clique as a basis for computational lower bounds. Planted clique has been used by Alon et al. (2007) as a basis for hardness for testing k-wise independence near the information theoretic limit and by Berthet and Rigollet (2013) to prove lower bounds for sparse principal component detection. Additionally, Hazan and Krauthgamer (2011) show that there is no PTAS for finding the best Nash equilibrium if finding a planted clique of size $\Theta(\log n)$ is not possible in polynomial time.

3.2 Reduction from planted clique to online local learning

We will reduce the planted clique problem to OLL, showing the following theorem:

Theorem 6. *If there exists a polynomial time online local learning algorithm that achieves $O(\sqrt{Nl^{1-\alpha}T})$ regret*[2] *for some $0 < \alpha < 1$, then there exists a polynomial time algorithm to distinguish between $G(n, 1/2)$ and $G(n, 1/2, k)$ for $k = n^{1/2-\epsilon}$ and $\epsilon = \frac{\alpha}{16}$ with probability $\frac{4}{5}$.*

Note that in the limit as $\epsilon \to 0$, we see that achieving $\sqrt{o(l)}$ dependence in the regret is not possible efficiently given Conjecture 1.

In fact, since our reduction is done in polynomial time, any algorithm for online local learning that does significantly better than the best algorithms for planted clique will lead to new algorithms for planted clique. Specifically, if we let $q = \max(N, l, T)$,

[2]We use N in this section to denote the size of S, i.e. the number of items that the online local learning algorithm must label with labels in $[l]$

then if we achieve an OLL algorithm that runs in time $q^{o(\log q)}$, we will obtain planted clique algorithms that run in time $n^{o(\log n)}$.

We sketch the proof here and give a full proof in Appendix B.1.

Proof Sketch. Given some G, we want to determine whether it is $G(n, 1/2)$ or $G(n, 1/2, k)$ for $k = n^{1/2-\epsilon}$. We construct an instance of OLL such that with probability $\frac{4}{5}$, the loss will be below some threshold in the planted case and above it in the non-planted case for an algorithm that achieves regret $O(\sqrt{Nl^{1-\alpha}T})$.

We first randomly partition G into $N = \frac{n}{l}$ clusters $C_1, ..., C_N$, each containing l vertices, where l is set such that $k = \Theta(\frac{n}{l} \log \frac{n}{l}) = \Theta(N \log N)$. The set of items for OLL will be the set of N clusters, and the label set size will still be l. Then we run $T = \binom{N}{2}$ rounds of OLL, one for each pair of clusters. For each cluster, we associate each of the l vertices of that cluster with one of the l labels of the label set. Note that this means that a fixed labeling of the clusters corresponds to picking one vertex from each cluster. Then in a given round of our reduction, when comparing two clusters (C_i, C_j), the loss function for a label pair (a, b) will be 0 if the vertex in C_i corresponding to label a has an edge to the vertex in C_j corresponding to label b and 1 if there is no such edge.

In the non-planted case, each edge exists independently with probability 1/2, so any algorithm's expected loss is $T/2$. Moreover, due to Chernoff bounds, we can get that the loss will be at least $T/4$ with high probability.

In the planted case, since there are N clusters and $N \log N$ clique vertices, each cluster will have at least one clique vertex with high probability. Then by picking one clique vertex from each cluster, we obtain a fixed labeling of the clusters that achieves 0 loss. Then by Markov, we can get that the loss in the planted case is $O(\sqrt{Nl^{1-\alpha}T})$ with probability $\frac{1}{c}$ for any constant c.

Simply plugging in the parameters shows that $\sqrt{Nl^{1-\alpha}T} \ll T/4$, which completes the proof. □

3.3 Planted dense subgraph

The planted dense subgraph problem is a generalization of the planted clique problem, making it an even stronger basis for computational lower bounds. In the planted dense subgraph problem, we must distinguish between two graphs on n vertices: $G(n,p)$ and $G(n,p,k,q)$. $G(n,p,k,q)$ is generated by taking a random subset of k vertices and putting edges between those vertices independently with probability q, then adding all other possible edges independently with probability p. One significant way in which the planted dense subgraph setting differs from planted clique is that the edge densities p and q are usually considered to be asymptotic functions of n and are much smaller than in the planted clique case. Moreover, a wider setting of the parameters p, q, k are not known to have polynomial time algorithms.

Unlike for planted clique, planted dense subgraph is not even known to have quasipolynomial time algorithms. We have the following conjecture:

Conjecture 2. *Given a graph G sampled either from $G(n,p)$ or $G(n,p,k,q)$, where $k = n^{1/2-\epsilon'}$ for $\epsilon' = \Omega(1)$ and $k = n^{\Omega(1)}$; $p = n^{-\alpha}$ for $\alpha = \Omega(1), \alpha \leq \frac{1}{2}$; $q = k^{-\alpha-\epsilon}$ for $\epsilon = \Omega(1)$; and $p = o(q)$, there is no polynomial time algorithm that can decide which ensemble G was sampled from with probability $\frac{4}{5}$.*[3]

The best algorithms for solving planted dense subgraph come from Bhaskara et al. (2010), who give an algorithm that can solve planted dense subgraph in time $n^{k^{\Theta(\epsilon)}}$, which for k polynomial in n gives a running time of $2^{n^{\Theta(\epsilon)}}$. Their algorithm involves

[3] Again, the constant is arbitrary and need only be bounded away from $\frac{1}{2}$.

counting certain constant-sized trees in the graph, which will appear with high probability only in the planted case. Note that their algorithm would run in exponential time for the parameters stated in Conjecture 2.

The planted dense subgraph problem is closely related to the Densest k-subgraph (DkS) problem, in which one attempts to find the densest subgraph of size k in some graph G. Experts believe that instances of planted dense subgraph present a natural barrier for improved approximation algorithms for DkS. Obtaining an efficient $o(n^{1/4})$ approximation for DkS would imply algorithms that can efficiently solve planted dense subgraph for certain parameter regimes. The best known efficient algorithms for approximating DkS are obtained by Bhaskara et al. (2010), who get an efficient $O(n^{1/4+\epsilon})$ approximation for DkS. Bhaskara et al. (2012) show an integrality gap for DkS of $\tilde{\Omega}(n^{1/4})$ for $\Omega(\frac{\log n}{\log \log n})$ rounds of the Sherali-Adams hierarchy, and their result applies in the parameter regimes of Conjecture 2. Similar conjectures have also been used to prove hardness of financial derivatives (Arora et al., 2010) and as a basis for public key cryptography (Applebaum et al., 2010).

3.4 Reduction from planted dense subgraph to online local learning

We give a reduction from planted dense subgraph to OLL that gives the following theorem:

Theorem 7. *Let $\epsilon = \Omega(1), \alpha = \Omega(1), k = n^{1/2-\epsilon'}$ for $\epsilon' = \Omega(1), k = n^{\Omega(1)}$ (that is, the parameters satisfy the conditions in Conjecture 2). If there exists a polynomial time*

algorithm that obtains regret[4] $O(\sqrt{Nl^\beta T})$ *for*

$$\beta = 2\frac{(1/2-\epsilon')(-\alpha-\epsilon+\frac{1}{2})-\omega\left(\frac{1}{\log n}\right)}{1/2+\epsilon'} \tag{3.4.1}$$

then one can distinguish $G(n,p)$ *and* $G(n,p,k,q)$ *with probability* $\frac{4}{5}$ *in polynomial time for* $p = n^{-\alpha}, k = n^{1/2-\epsilon'}, q = k^{-\alpha-\epsilon}$.

As in the planted clique case, we can see that in the limit as $\epsilon, \alpha, \epsilon' \to 0$, we have that achieving $\sqrt{o(l)}$ dependence in the regret is not possible efficiently given Conjecture 2.

As in the planted clique case, since our reduction is done in polynomial time, any algorithm for online local learning that does significantly better than the best algorithms for planted dense subgraph will lead to new algorithms for planted dense subgraph. Namely if we let $q = \max(N, l, T)$, then if we achieve an OLL algorithm that runs in time $2^{q^{o(1)}}$, we will obtain planted dense subgraph algorithms that run in time $2^{n^{o(1)}}$, which is better than the current state of the art.

We sketch the proof here, which mirrors the construction for the planted clique setting. The full proof is in Appendix B.2.

Proof sketch. The reduction to planted dense subgraph is essentially the same as for planted clique. Given some G, we want to determine whether it is $G(n,p)$ or $G(n,p,k,q)$ for appropriate settings of the parameters p, k, q. We will use payoffs instead of losses in this section for convenience. The setup is entirely equivalent, and regret is defined as the difference between the payoff of the best fixed labeling and the expected payoff of the algorithm. We construct an instance of OLL such that

[4] As in the planted clique section, we use N to denote the size of S, i.e. the number of items that the online local learning algorithm must label with labels in $[l]$

with probability $\frac{4}{5}$, the payoff will be above some threshold in the planted case and below it in the non-planted case for an algorithm that achieves regret $O(\sqrt{Nl^\beta T})$.

We first randomly partition G into $N = \frac{n}{l}$ clusters $C_1, ..., C_N$, each containing l vertices, where l is set such that $k = 10N$.[5] The set of items for OLL will be the set of N clusters, and the label set size will still be l. Then we run $T = \binom{N}{2}$ rounds of OLL, one for each pair of clusters. For each cluster, we associate each of the l vertices with one of the l labels. Note that this means that a fixed labeling corresponds to picking one vertex from each cluster. Then in a given round of our reduction, when comparing two clusters (C_i, C_j), the payoff function for a label pair (a, b) will be 1 if the vertex in C_i corresponding to label a has an edge to the vertex in C_j corresponding to label b and 0 if there is no such edge.

In the non-planted case, each edge exists with independently probability p, so any algorithm's expected payoff is pT. Moreover, due to Chernoff bounds, we can get that the payoff will be at most $pT + 5\sqrt{pT}$ with high probability.

In the planted case, consider the event that cluster i does not have a clique vertex. This occurs with probability $(1 - \frac{1}{N})^k = (1 - \frac{1}{N})^{10N} \leq e^{-10}$. Then the total number of clusters without a clique vertex will be below some constant fraction of N with good probability due to linearity of expectation and Markov. Thus, we know that cN of the clusters have a clique vertex for some constant c, so a fixed labeling that includes these clique vertices will get expected payoff q over $\binom{cN}{2}$ rounds. Then the expected payoff of the algorithm is at least:

$$q\binom{cN}{2} - O(\sqrt{Nl^\beta T}) \tag{3.4.2}$$

[5] This setting of k gives a slightly tighter reduction than setting $k = N \log N$. A similar change in the planted clique case would also yield a better dependence on α for ϵ.

We can use Markov to show that with good probability, the algorithm's performance will be no less than $q\binom{cN}{2} - O(\sqrt{Nl^\beta T})$. Then since $p = o(q)$, we simply need to show that $\sqrt{Nl^\beta T} \ll q\binom{cN}{2}$ to get that the payoff of the algorithm will be asymptotically different in the planted versus non-planted cases. Plugging in the settings for each parameter gives the result. □

Chapter 4

Applications and related work

4.1 Regret bounds for specific online learning problems

4.1.a Online max cut

In the online max cut setting, in each round the learner receives a pair of vertices $(v_i, v_j) \in V$, where $|V| = n$ and must predict "cut" or "not cut". Then nature gives an answer of "cut" or "not cut" and assigns loss accordingly. The goal is to achieve low regret with respect to the best cut in the graph.

We can model this as an online local learning problem for $l = 2$, i.e. we label each vertex according to what side of the cut it is on. Our predictions are a probability distribution over four outcomes, where assigning both vertices the same label corresponds to a prediction of "not cut" and assigning different labels corresponds to a prediction of "cut". Then our regret in this case is $O(\sqrt{nT})$, which is optimal for any online max cut algorithm.

4.1.b Online gambling

In the online gambling setting, in each round the learner receives a pair of teams (x_i, x_j) from some set of n teams and the learner must predict which team will win, after which nature provides the answer. The goals is to achieve low regret with respect to the best fixed permutation of the teams, i.e. the i^{th} team will beat the j^{th} team iff $i < j$.

We can model this as an online local learning problem for $l = n$, where we label each team with a number in $[n]$, so that given a pair of teams (x_i, x_j), the label pair (a, b) corresponds to i winning iff $a < b$. In this case, we obtain regret $O(\sqrt{n^2 T})$. This is considerably worse than the $\Omega(\sqrt{n \log(n) T})$ lower bound from Kleinberg et al. (2010) and the $O\left(\sqrt{n \log^3(n) T}\right)$ regret achieved by Hazan et al. (2012). The Hazan et al. (2012) algorithm is discussed in more detail in section 4.2.

4.1.c Online collaborative filtering

In the online collaborative filtering setting, there is a set of n users and m items. In each round, the learner receives a pair $(x, y) \in [n] \times [m]$ and must make a prediction in $\{0, 1\}$, which represents whether user x likes item y or not.

With no further restrictions, this problem would be equivalent to doing nm separate prediction problems. However, people generally assume that similar users will like similar movies, which should aid in prediction. To enforce this assumption, it is often assumed that the $n \times m$ matrix being learned has low trace norm. An alternate assumption could be that users are drawn from a set of k_1 types, while movies are drawn from a set of k_2 types, where we assume k_1 and k_2 are much smaller than n and m. Then to model this as an OLL problem, our set S will consist of the k_1 types of users and the k_2 types of movies, so $|S| = k_1 + k_2$. We will have two labels,

corresponding to liking or not liking, and the loss will just be 1 for the incorrect label and 0 for the correct label. This gives regret $O(\sqrt{(k_1 + k_2)T})$ compared to a perfect labeling.

4.2 Related work

The OLL framework closely resembles previous work by Hazan et al. (2012), which examines online matrix prediction (OMP) problems. The online matrix prediction setting proceeds in rounds, and in each round the learner receives some pair $(i_t, j_t) \in [m] \times [n]$ and gives a prediction in $[-1, 1]$. The learner then receives a loss function $l_t : [-1, 1] \to \mathbb{R}$. In this framework, the learner competes against a comparison class of matrices $\mathcal{W} \subseteq [-1, 1]^{m \times n}$. In particular, the authors focus on classes of matrices that are (β, τ)-decomposable, for which they give an algorithm that achieves $\tilde{O}(\sqrt{\beta\tau T})$ regret[1] for any class of (β, τ)-decomposable matrices. A matrix M is (β, τ)-decomposable if $sym(M) = P - N$,[2] where P and N are PSD, the diagonal entries of P and N are bounded by β, and $\text{tr}(P) + \text{tr}(N) \leq \tau$. Roughly, finding a good (β, τ)-decomposable class of matrices can be thought of as finding a comparison class of matrices with low trace norm.

Both the OLL framework and the OMP framework can be used to solve similar problems, and neither is strictly better than the other. For online max cut, the OLL framework achieves the optimal $O(\sqrt{nT})$ regret because there are a constant number of labels, while the OMP framework achieves $O(\sqrt{nT \log n})$ regret, which is tight for the framework.

[1] The $\tilde{O}$ suppresses polylog factors

[2] $sym(M)$ is defined as M for symmetric M and as the matrix $\left(\begin{smallmatrix} 0 & M \\ M^\intercal & 0 \end{smallmatrix}\right)$ for non symmetric M.

Hazan et al. (2012) use a Matrix Multiplicative Weights (MMW) algorithm, which can be thought of as doing FTRL with the von Neumann entropy as a regularizer. In the case of online max cut, the log determinant appears to be a better regularizer.

For the case of online gambling, the OLL framework achieves only $O(\sqrt{n^2T})$ regret at best, while the MMW algorithm that achieves $O\left(\sqrt{n\log^3 nT}\right)$ regret. The poor performance of OLL is due to the fact that n labels are used, which essentially corresponds to using a hypothesis class that is larger than necessary. On the other hand, Hazan et al. (2012) found a more succinct class of hypothesis matrices that are $(\log n, n\log n)$-decomposable, leading to the improved performance bounds.

For the online collaborative filtering setting, the OLL framework has no way to leverage the usual assumption of low trace norm for the predicted matrix. The algorithm by Hazan et al. (2012) achieves $O(\sqrt{\tau n^{1/2}\log(n)T})$ regret for matrices of trace norm at most τ, where it is assumed that $m \leq n$ without loss of generality. The OLL framework can function under a different assumption of a limited set of k_1 types of users and k_2 types of movies, which gives regret $O(\sqrt{(k_1+k_2)T})$.

In general, to use the OMP framework, one must find an appropriate (β, τ)-decomposable class of matrices. In problems where a good (β, τ)-decomposable class can be found, the MMW algorithm does well. On the other hand, in the online local learning framework, the hypothesis class is derived in a simple way from the problem statement. As such, the OLL algorithm performs poorly for certain problems where important structure is ignored. Nonetheless, for problems that fall naturally into a labeling formulation, OLL provides a simple way to obtain good regret.

Chapter 5

Open Problems

There are several open problems relating to OLL. OLL deals with predictions over pairs of items–this may be expanded to predictions over r-tuples of items. Moreover, our lower bounds do not preclude inefficient algorithms that achieve better regret. Finally, it is still unclear how to reconcile the difference in performance of the log determinant compared to other regularizers on various problems.

5.1 Predictions over r-tuples

In the OLL framework as stated, the learner receives pairs of items in each round. One might want to extend this framework to cover r-tuples of items. This can be done trivially with the current OLL algorithm by considering sets of size $\lceil \frac{r}{2} \rceil$ as items, thus increasing the item set size to $n^{\lceil \frac{r}{2} \rceil}$ and increasing the label set size to $l^{\lceil \frac{r}{2} \rceil}$, which will give regret $O(\sqrt{n^{\lceil \frac{r}{2} \rceil} l^{\lceil \frac{r}{2} \rceil} T})$. In the case of constant size label set, we get regret $O(\sqrt{n^{\lceil \frac{r}{2} \rceil} T})$.

Christiano (2014b) shows a lower bound in this setting, namely that a constant number of rounds of the Lasserre hierarchy of SDP relaxations cannot achieve regret

$o(\sqrt{n^{\frac{r}{2}}T})$ for a constant size label set. Moreover, he shows that an algorithm that achieves regret $O(\sqrt{cT})$ would be able to solve planted constraint satisfaction problems using c clauses. Since the current best algorithms require $n^{\frac{r}{2}}$ clauses to solve planted constraint satisfaction problems, this gives further evidence that $o(\sqrt{n^{\frac{r}{2}}T})$ regret may be intractable (Feldman et al., 2013b).

5.2 Subexponential algorithms for online local learning

The lower bounds we provided for OLL are based on hardness of planted clique and planted dense subgraph. However, there exist subexponential algorithms for planted dense subgraph, which means that there could exist subexponential time algorithms that solve OLL with better regret than $O(\sqrt{nlT})$. We know that running the exponential time experts algorithm gives $O(\sqrt{n\log(l)T})$ regret, but it is open as to whether there exist subexponential time algorithms that have better regret than the polynomial time algorithms for OLL.

5.3 Combining regularizers

Hazan et al. (2012) obtain close to optimal regret for the online gambling problem using the von Neumann entropy as a regularizer. It is open as to what the best regularizer is for particular online prediction problems. Hazan et al. (2012) and Christiano (2014a) also optimize over different convex bodies, which are tuned to be appropriate for the regularizers they chose respectively. Nonetheless, it is possible that their two frameworks can be combined somehow to obtain better regret, or

at least to obtain optimal regret on a wider array of problems using some unified algorithm.

Chapter 6

Conclusion

We presented matching upper and lower bounds for the online local learning setting, establishing the capabilities and limitations for algorithms in this setting. The upper bound involved improving the analysis of Christiano's algorithm using the Follow-the-Regularized-Leader framework. The lower bound relied on a reductions from the planted clique problem and the planted dense subgraph problem. We presented hardness conjectures for these two problems as well as evidence that these conjectures are true. We also compared the online local learning framework to the related online matrix prediction setting by Hazan et al. (2012), which achieves better regret bounds for certain problems. Finally, we discussed several open problems remain in the online local learning framework.

Bibliography

Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1988.

Noga Alon, Alexandr Andoni, Tali Kaufman, Kevin Matulef, Ronitt Rubinfeld, and Ning Xie. Testing k-wise and almost k-wise independence. In *Proceedings of the 39th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 496–505, 2007.

Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 171–180. ACM, 2010.

Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. In *ICS*, pages 49–65, 2010.

Pranjal Awasthi, Moses Charikar, Kevin A. Lai, and Andrej Risteski. Label optimal regret bounds for online local learning. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, 2015.

Quentin Berthet and Philippe Rigollet. Complexity theoretic lower bounds for sparse principal component detection. *Journal of Machine Learning Research, W and CP*, 30:1046–1066, 2013.

Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 201–210. ACM, 2010.

Aditya Bhaskara, Moses Charikar, Aravindan Vijayaraghavan, Venkatesan Guruswami, and Yuan Zhou. Polynomial integrality gaps for strong SDP relaxations of densest k-subgraph. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 388–405. SIAM, 2012.

Paul Christiano. Online local learning via semidefinite programming. In *Proceedings of the 46th Annual ACM Symposium on the Theory of Computing (STOC)*, 2014a.

Paul Christiano. Open problem: Online local learning. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 1290–1294, 2014b.

Yael Dekel, Ori Gurel-Gurevich, and Yuval Peres. Finding hidden cliques in linear time with high probability. *Combinatorics, Probability and Computing*, 23:29–49, 2014.

Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh Vempala, and Ying Xiao. Statistical algorithms and a lower bound for detecting planted cliques. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 655–664. ACM, 2013a.

Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted solutions. 2013b. URL http://arxiv.org/abs/1311.4821.

Elad Hazan. The convex optimization approach to regret minimization. Technical report, 2009.

Elad Hazan and Robert Krauthgamer. How hard is it to approximate the best nash equilibrium? *SIAM Journal on Computing*, 40(1):79–91, 2011.

Elad Hazan, Satyen Kale, and Shai Shalev-Shwartz. Near-optimal algorithms for online matrix prediction. In *The 25th Conference on Learning Theory (COLT)*, 2012.

Mark Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–360, 1992.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2):245–272, 2010.

Ludek Kucera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Jan R. Magnus and Heinz Neudecker. Matrix differential calculus with applications in statistics and econometrics. 1995.

Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In *Proceedings of the 47th Annual ACM Symposium on the Theory of Computing (STOC)*, 2015.

Appendix A

Upper bound proofs

A.1 Sketch of Christiano (2014a) bound on λ

Here we sketch the bound on the strong convexity parameter λ in Christiano (2014a). Recall that λ is defined as follows:

$$\lambda = \max_{t, x \in \mathcal{K}} f_t^\intercal [\nabla^2 \mathcal{R}(x)]^{-1} f_t \tag{A.1.1}$$

Christiano (2014a) proves that $\lambda \leq l^2$, where l is the size of the label set in the OLL setting.

For his approach, he treats the regularizer as concave instead of convex and uses the following modified version of Theorem 1, which uses an alternate characterization of strong concavity, where γ is equivalent to $\frac{1}{\lambda}$:

Theorem 8 (Christiano (2014a))**.** *Suppose that $\mathcal{R}$ is a strongly concave regularizer bounded between 0 and D on some convex body $\mathcal{K}$. Moreover, suppose that whenever*

$|Loss_t(A) - Loss_t(A')| \geq \delta$ *for* $A, A' \in \mathcal{K}$*:*

$$\mathcal{R}(\epsilon A + (1-\epsilon)A') \geq \epsilon\mathcal{R}(A) + (1-\epsilon)\mathcal{R}(A') + \epsilon(1-\epsilon)\gamma\delta^2 \tag{A.1.2}$$

Then FTRL using $\mathcal{R}$ *as a regularizer over* $\mathcal{K}$ *achieves regret* $O\left(\sqrt{\frac{DT}{\gamma}}\right)$.

His proof consists of several parts. Throughout, he uses the interpretation of $\log\det(\Sigma)$ as the differential entropy of a Gaussian with covariance matrix Σ. Let us define the Gaussians G_1 and G_2 with covariance matrices $\Sigma_1 = lA+I$ and $\Sigma_2 = lA'+I$. First, he argues that if $|\text{Loss}_t(A) - \text{Loss}_t(A')| \geq \delta$, then one of the second-order moments will differ substantially between G_1 and G_2, which he shows implies that the total variation distance between G_1 and G_2 is also large, namely $\Omega(\frac{\delta}{l})$. Then he uses a standard strong concavity result about the differential entropy that shows that given probability distributions P and Q with total variation distance Δ,

$$H(\epsilon P + (1-\epsilon)Q) \geq \epsilon H(P) + (1-\epsilon)Q + \epsilon(1-\epsilon)\Delta^2 \tag{A.1.3}$$

where H is the differential entropy. Using this result for $\Delta = \frac{\delta}{l}$ gives the condition in Theorem 8 for $\gamma = \frac{1}{l^2}$.

A.2 Proofs for lemmas in section 2.3

For completeness, we replicate the bound on the diameter of D from Christiano (2014a).

Lemma 4. *For the log determinant regularizer* $\mathcal{R}$ *over* $\mathcal{K}$*, we have* $D \leq nl$*, where*

$$D = \max_{x_1, x_2 \in \mathcal{K}} |\mathcal{R}(x_1) - \mathcal{R}(x_2)|$$

as defined in Theorem 1.

Proof. Since all $A \in \mathcal{K}$ are positive semidefinite, the eigenvalues of $lA + I$ are all at least 1, so $\det(lA+I) \geq 1$ and $\log\det(lA+I) \geq 0$. Next, we know that $\mathrm{tr}(A) \leq n$, so $\mathrm{tr}(lA+I) \leq 2nl$. Given this trace condition, $\det(lA+I)$ is maximized if all eigenvalues are equal, that is, if $lA = I$. Equivalently, we could use the fact $\log\det(B) \leq \mathrm{tr}(B-I)$ for any PSD matrix B. Then $\log\det(lA + I) \leq \log\det(2I) = \log(2^{nk}) = nl$. □

Lemma 5. *The inverse Hessian H^{-1} of the log determinant regularizer has the following form:*

$$H^{-1}_{(w,x),(y,z)} = -l^{-2}(lA + I)_{y,x}(lA + I)_{w,z}$$

for all $x, w, y, z \in S \times L$.

Before we get into the proof of Lemma 5, we will need to prove the following lemma:

Lemma 9. *Let $H = \nabla^2\mathcal{R}(A)$ be the Hessian of the log determinant regularizer $\mathcal{R}(A)$. Then H has the following form:*

$$H_{(w,x),(y,z)} = -l^2(lA + I)^{-1}_{x,y}(lA + I)^{-1}_{z,w}$$

for all $w, x, y, z \in S \times L$.

Proof. Let $X = lA + I, Z = \det X$, and $w, x, y, z \in S \times l$. Then we can compute the first order partial derivatives:

$$\frac{\partial\mathcal{R}(A)}{\partial A_{w,x}} = \frac{1}{Z}\frac{\partial Z}{\partial A_{w,x}} \tag{A.2.1}$$

Now we use Jacobi's formula (eg. from Magnus and Neudecker (1995)):

Lemma 10 (Jacobi's formula)**.** $\frac{\partial \det(B)}{\partial B_{i,j}} = \det(B) B^{-1}_{j,i}$

Plugging this into (A.2.1) gives:

$$\frac{\partial \mathcal{R}(A)}{\partial A_{w,x}} = \frac{1}{Z}(Z X^{-1}_{x,w}) l = l X^{-1}_{x,w} \tag{A.2.2}$$

Next, we compute the entries of the Hessian:

$$\frac{\partial^2 \mathcal{R}(A)}{\partial A_{w,x} \partial A_{y,z}} = l \frac{\partial X^{-1}_{x,w}}{\partial A_{y,z}} \tag{A.2.3}$$

We use the fact that for an invertible matrix B, $\frac{\partial B^{-1}}{\partial t} = -B^{-1} \frac{\partial B}{\partial t} B^{-1}$. Thus,

$$\frac{\partial X^{-1}_{x,w}}{\partial A_{y,z}} = -\left(X^{-1} \frac{\partial X}{\partial A_{y,z}} X^{-1} \right)_{x,w} \tag{A.2.4}$$

Let $Y = \frac{\partial X}{\partial A_{y,z}}$. Then,

$$\frac{\partial^2 \mathcal{R}(A)}{\partial A_{w,x} \partial A_{y,z}} = -l (X^{-1} Y X^{-1})_{x,w} = -l \sum_p (X^{-1} Y)_{x,p} X^{-1}_{p,w} \tag{A.2.5}$$

$$= -l \sum_p \left(\sum_q X^{-1}_{x,q} Y_{q,p} \right) X^{-1}_{p,w} \tag{A.2.6}$$

Then since the only nonzero entry of Y is $Y_{y,z} = l$, we get:

$$\frac{\partial^2 \mathcal{R}(A)}{\partial A_{w,x} \partial A_{y,z}} = -l^2 X^{-1}_{x,y} X^{-1}_{z,w} \tag{A.2.7}$$

□

Proof of Lemma 5. Let $w, x, y, z \in S \times L$ and let $X = lA + I$. To prove the lemma, we simply need to verify that $H^{-1} H = I$, which means that $(H^{-1} H)_{(w,x),(y,z)} = \delta_{(w,x),(y,z)}$,

where $\delta_{a,b} = 1$ if $a = b$ and 0 otherwise.

$$(H^{-1}H)_{(w,x),(y,z)} = \sum_{p,q} H^{-1}_{(w,x),(p,q)} H_{(p,q),(y,z)} \tag{A.2.8}$$

$$= \sum_{p,q} \left(-l^{-2} X_{p,x} X_{w,q}\right) \left(-l^{2} X^{-1}_{q,y} X^{-1}_{z,p}\right) \tag{A.2.9}$$

$$= \sum_{p} X^{-1}_{z,p} X_{p,x} \sum_{q} X_{w,q} X^{-1}_{q,y} \tag{A.2.10}$$

$$= \delta_{x,z}\delta_{w,y} = \delta_{(w,x),(y,z)} \tag{A.2.11}$$

□

Appendix B

Lower bound proofs

B.1 Planted clique lower bound proof

Theorem 6. *If there exists a polynomial time online local learning algorithm that achieves $O(\sqrt{Nl^{1-\alpha}T})$ regret*[1] *for some $0 < \alpha < 1$, then there exists a polynomial time algorithm to distinguish between $G(n, 1/2)$ and $G(n, 1/2, k)$ for $k = n^{1/2-\epsilon}$ and $\epsilon = \frac{\alpha}{16}$ with probability $\frac{4}{5}$.*

Proof. We are given a graph G and an OLL algorithm A that achieves $O(\sqrt{Nl^{1-\alpha}T})$ regret, and we must now determine if G is $G(n, 1/2)$ or $G(n, 1/2, k)$ for $k = n^{1/2-\epsilon}$. We will construct an instance of OLL such that with probability at least $\frac{4}{5}$ the following holds: we can find a threshold such that if the loss is below that threshold, then the graph is $G(n, 1/2, k)$, otherwise it is $G(n, 1/2)$.

We first partition the vertices of G into $N = \frac{n}{l}$ clusters $C_1, ..., C_N$, each of size l, where l is chosen such that $k = c_k \frac{n}{l} \log \frac{n}{l} = c_k N \log N$ for some constant c_k.[2] For each cluster C_i, number the vertices in that cluster arbitrarily as $v_1^i, ..., v_l^i$. We will now

[1] We use N in this section to denote the size of S, i.e. the number of items that the online local learning algorithm must label with labels in $[l]$

[2] We can assume wlog that l divides n.

play $T = \binom{N}{2}$ rounds of OLL, one for each pair of clusters (C_i, C_j). In each round, we give A a pair (C_i, C_j) and the loss for a pair of labels (a, b) is 0 if (v_a^i, v_b^j) is an edge in G and 1 otherwise.

In the non-planted case, $G = G(n, 1/2)$. Then for all $i, j \in [N]$ and $a, b \in [l]$, the edge (v_a^i, v_b^j) is in the graph with probability $1/2$, so any algorithm will have expected loss of $T/2$. Moreover, the algorithm's loss is simply the sum of T coin flips, so by Chernoff bounds we get:

$$\Pr[\text{Loss} \leq T/2 - T/4] \leq e^{-\frac{2(T/4)^2}{T}} = e^{-\frac{T}{8}} \tag{B.1.1}$$

So the algorithm's loss is at least $T/4$ with high probability.

Now consider the planted case. We have N clusters and $\Theta(N \log N)$ planted clique vertices. Since the partition was chosen randomly, a coupon collector argument implies that with high probability, each cluster will contain at least one clique vertex. Call this event E_0. If E_0 occurs, then for each cluster C_i, we can identify a representative clique vertex $v_{a_i}^i$ contained in C_i (if there are multiple clique vertices in C_i, pick one arbitrarily). If we consider the clique vertices $\{v_{a_1}^1, ..., v_{a_N}^N\}$, we have a fixed labeling, corresponding to labeling cluster C_i with label a_i. This labeling will have 0 loss over the $\binom{N}{2}$ rounds of our reduction.

Since A achieves regret $O(\sqrt{Nl^{1-\alpha}T})$, we know that if E_0 occurs, then A's expected loss in the planted case is at most $c_l\sqrt{Nl^{1-\alpha}T}$ for some constant $c_l > 0$. Moreover, by Markov's inequality, we get:

$$\Pr[\text{Loss}(A) \geq 10c_l\sqrt{nl^{1-\alpha}T}] \leq \frac{1}{10} \tag{B.1.2}$$

Then to distinguish between the two cases with probability $\frac{4}{5}$, we simply need the loss in each case to be different asymptotically with probability at least $\frac{4}{5}$. As we have shown, the loss in the planted case is at most $10c_l\sqrt{Nl^{1-\alpha}T}$ with probability at least $\frac{1}{10}$, while the loss in the non-planted case is at least $T/4$ with high probability. Then we want the following to hold:

$$10c_l\sqrt{Nl^{1-\alpha}T} = o(T/4) \tag{B.1.3}$$

$$\iff Nl^{1-\alpha} = o(T) \tag{B.1.4}$$

Let us look at the left side of this equation. We substitute parameters to get everything in terms of N. Since $k = n^{1/2-\epsilon}$ and $k = c_k N \log N$, we get $n = (c_k N \log N)^{\frac{1}{1/2-\epsilon}}$ and $l = \frac{n}{N} = \frac{(c_k N \log N)^{\frac{1}{1/2-\epsilon}}}{N}$. Then the left side of (B.1.4) becomes:

$$N\left(\frac{(c_k N \log N)^{\frac{1}{1/2-\epsilon}}}{N}\right)^{1-\alpha} = N^{\alpha}(N\log N)^{\frac{1-\alpha}{1/2-\epsilon}} \leq N^{\alpha}N^{(1-\alpha)(2+8\epsilon)}(\log^c N) \tag{B.1.5}$$

for $c = 2(1+4\epsilon)$, where the inequality is due to the fact that $\frac{1}{1/2-\epsilon} \leq 2(1+4\epsilon)$ for $\epsilon < 1/2$. Then we substitute in $\epsilon = \frac{\alpha}{16}$ to get:

$$N^{(1-\alpha)(2+\alpha/2)+\alpha}(\log^c N) = N^{2-\alpha/2-\alpha^2/2}(\log^c N) \tag{B.1.6}$$

Since $\alpha > 0$ and $T = \binom{N}{2} = \Theta(N^2)$, we know that (B.1.6) is $o(T)$, which completes the proof. □

B.2 Planted dense subgraph lower bound proof

Theorem 7. *Let $\epsilon = \Omega(1), \alpha = \Omega(1), k = n^{1/2-\epsilon'}$ for $\epsilon' = \Omega(1), k = n^{\Omega(1)}$ (that is, the parameters satisfy the conditions in Conjecture 2). If there exists a polynomial time*

algorithm that obtains regret[3] $O(\sqrt{Nl^\beta T})$ *for*

$$\beta = 2\frac{(1/2-\epsilon')(-\alpha-\epsilon+\frac{1}{2})-\omega\left(\frac{1}{\log n}\right)}{1/2+\epsilon'} \tag{B.2.1}$$

then one can distinguish $G(n,p)$ *and* $G(n,p,k,q)$ *with probability* $\frac{4}{5}$ *in polynomial time for* $p = n^{-\alpha}$ *and* $q = k^{-\alpha-\epsilon}$.

Proof. This proof will closely mirror the setup for the planted clique case. We are given a graph G and an OLL algorithm A that achieves $O(\sqrt{Nl^\beta T})$ regret, and we must now determine if G is $G(n,p)$ or $G(n,p,k,q)$ for $k = n^{1/2-\epsilon'}$.

We will use payoffs instead of losses in this section for convenience. The setup for payoffs is entirely equivalent to the setup for losses, and regret is defined as how much less an algorithm's expected payoff is compared to the payoff of the best fixed labeling.

We will construct an instance of OLL such that with probability at least $\frac{4}{5}$ the following holds: we can find a threshold such that if the payoff is above that threshold, then the graph is $G(n,p,k,q)$, otherwise it is $G(n,p,k,q)$.

We first partition the vertices of G into $N = \frac{n}{l}$ clusters $C_1, ..., C_N$, each of size l, where l is chosen such that $k = 10N$.[4] For each cluster C_i, number the vertices in that cluster arbitrarily as $v_1^i, ..., v_l^i$. We will now play $T = \binom{N}{2}$ rounds of OLL, one for each pair of clusters (C_i, C_j). In each round, we give A a pair (C_i, C_j) and the payoff for a pair of labels (a,b) is 1 if (v_a^i, v_b^j) is an edge in G and 0 otherwise.

In the non-planted case, $G = G(n,p)$. Then for all $i,j \in [N]$ and $a,b \in [l]$, the edge (v_a^i, v_b^j) is in the graph with probability p, so any algorithm will have expected

[3] As in the planted clique section, we use N to denote the size of S, i.e. the number of items that the online local learning algorithm must label with labels in $[l]$

[4] We use $k = \Theta(N)$ instead of $k = \Theta(N \log N)$ because the former achieves a slightly tighter reduction in the parameters of the problem. One could make a similar change to the proof for planted clique to get a slightly better dependence on α for ϵ.

payoff of pT. Moreover, the algorithm's payoff is simply the sum of T coin flips, so by Chernoff bounds we get that the payoff is at most $pT + 5\sqrt{pT}$ with high probability.

Now consider the planted case. We have N clusters and $\Theta(N)$ planted clique vertices. Let $\mathcal{I}_i$ be the indicator random variable that is 1 if a clique vertex does not fall in cluster i and 0 otherwise. Then we have:

$$\mathrm{E}\left[\sum_{i=1}^{N} \mathcal{I}_i\right] = N\left(1 - \frac{1}{N}\right)^k = N\left(1 - \frac{1}{N}\right)^{10N} \leq Ne^{-10} \tag{B.2.2}$$

Then by Markov, we get:

$$\Pr\left[\sum_{i=1}^{N} \mathcal{I}_i \geq \frac{N}{10}\right] \leq \frac{Ne^{-10}}{\frac{N}{10}} \leq \frac{1}{10} \tag{B.2.3}$$

Thus, with probability $\frac{9}{10}$, at least $\frac{9}{10}N$ clusters contain a clique vertex. Call this event E_0. In this case, a fixed labeling that uses all of these clique vertices will have expected payoff q over $\binom{9N/10}{2}$ rounds of our reduction.

Let OPT be the payoff of the best fixed labeling. We know the algorithm's regret is at most $c_l\sqrt{Nl^\beta T}$ of OPT for some constant $c_l > 0$. Let us now show that with probability $\frac{9}{10}$, the algorithm has payoff at least $OPT - 10c_l\sqrt{Nl^\beta T}$. Call this event E_1. Let $X = OPT - \text{Payoff}(A)$. Then,

$$\mathrm{E}[X] = OPT - \mathrm{E}[\text{Payoff}(A)] = \text{Regret} \leq c_l\sqrt{Nl^\beta T} \tag{B.2.4}$$

We also know that $\mathrm{E}[X] = \mathrm{E}[X|X \geq 0] + \mathrm{E}[X|X < 0]$. Payoff$(A)$ can only be less than OPT for $X \geq 0$. Thus, if we use $\mathrm{E}[X|X \geq 0]$ in Markov's inequality, we get:

$$\Pr[X \geq 10c_l\sqrt{Nl^\beta T}|X \geq 0] \leq \frac{1}{10} \tag{B.2.5}$$

Thus, with probability at least $\frac{81}{100}$, E_0 and E_1 occur, so the algorithm gets payoff at least $q\binom{9N/10}{2} - 10c_l\sqrt{Nl^\beta T}$. Then to distinguish between the planted and non-planted cases with probability $\frac{4}{5}$, we simply need the payoff in each case to be different asymptotically with probability at least $\frac{4}{5}$. That is, we want the following to hold:

$$pT + 5\sqrt{pT} \ll q\binom{9N/10}{2} - 10c_l\sqrt{Nl^\beta T} \tag{B.2.6}$$

Since $p = o(q)$ and $T = \binom{N}{2}$, we simply need to show:

$$\sqrt{Nl^\beta N^2} \ll o(qN^2) \tag{B.2.7}$$

Or alternately, we can show:

$$l^{\beta/2} \ll o(q\sqrt{N}) \tag{B.2.8}$$

Let us look at the left side of this equation. We substitute parameters to get everything in terms of n. Since $k = 10N$ and $k = n^{1/2-\epsilon'}$, we get $l = \frac{n}{N} = \frac{10n}{n^{1/2-\epsilon'}} = 10n^{1/2+\epsilon'}$. Meanwhile, $q = k^{-\alpha-\epsilon} = n^{(1/2-\epsilon')(-\alpha-\epsilon)}$. (B.2.8) is true if:

$$\left(n^{1/2+\epsilon'}\right)^{\beta/2} = o\left(n^{(1/2-\epsilon')(-\alpha-\epsilon)}n^{(1/2-\epsilon')/2}\right) \tag{B.2.9}$$

Then since $n^{\omega(\frac{1}{\log n})} = \omega(1)$, it suffices to show:

$$(1/2+\epsilon')\frac{\beta}{2} = (1/2-\epsilon')(-\alpha-\epsilon) + \frac{1}{2}(1/2-\epsilon') - \omega\left(\frac{1}{\log n}\right) \tag{B.2.10}$$

which is easily shown to be true for our setting of β. □